INF2080

Regular Expressions

Daniel Lupp

Universitetet i Oslo

28.01.2015

Department of Informatics

University of Oslo

Group session tomorrow

There will be no group session tomorrow, Jan. 29. Friday (Jan 30) there will be a group session as planned.

Regular Expressions

Definition (Regular Expression)

Given an alphabet Σ, a *regular expression* is

- a for some $a \in \Sigma$,
- ε,
- $\emptyset$,
- $(R_1 \cup R_2)$ for regular expressions R_1, R_2,
- $(R_1 R_2)$ for regular expressions R_1, R_2,
- R_1^* for a regular expression R_1.

$\rightarrow$ Regular expressions represent languages!

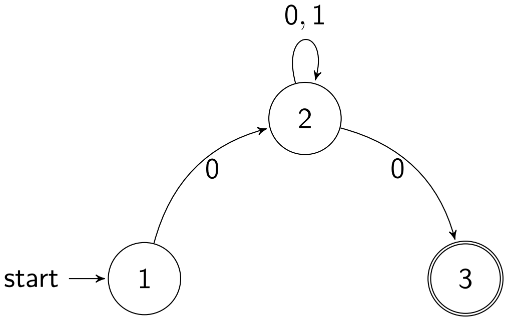
Regular Expressions - Examples

What languages do the following regular expressions (RE) represent?

- 0^*
- 10^*1
- $(1(0 \cup 1)^*1) \cup (0(0 \cup 1)^*0) \cup 0 \cup 1$

What is the connection between RE and DFA/NFA?

Language $0(0 \cup 1)^*0$:



Regular Expressions and Automata

What is the connection between RE and DFA/NFA?

- Can all RE be represented using DFA/NFA?
- Can all DFA/NFA be described by RE?

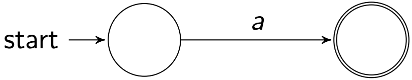
Last Lecture: Yes! Now some examples.

Regular Expressions and Automata

Proposition

Every language described by an RE is regular.

Proof based on inductive definition of RE, e.g.,: if $R = a$ for $a \in \Sigma$, then the corresponding language $L(R) = \{a\}$ is accepted by the following DFA:



Rest of the proof is based on unions, concatanations, and Kleene stars of languages and corresponding DFAs, see exercises and the book.

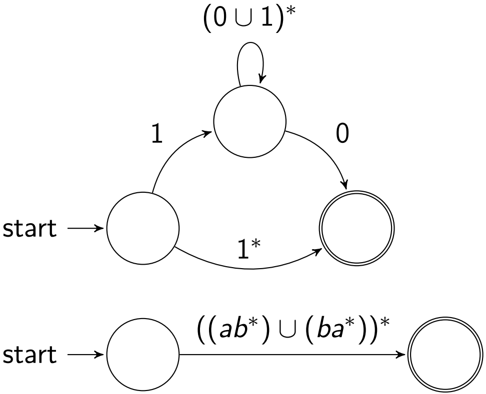
Proposition

Every regular language can be described using a RE.

GNFA

Generalized Nondeterministic Finite Automaton (GNFA):

- NFA where the transitions are RE, not only symbols from Σ.
- some other assumptions for convenience:
- start state goes to every other state, but has no incoming states
- every state goes to the unique accepting state, which is different from the starting state. The accepting state does not have any outgoing arrows.
- all other states have one transition to all other states, including themselves.



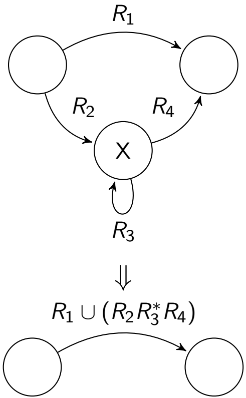
Proposition

Every regular language can be described using a RE.

Proof idea: take DFA and transform into a GNFA that accepts the same language. Iteratively remove (non-starting and non-final) states so that the same language is accepted, until only the starting and accepting state remain. Then the RE along the transition between the two states describe the regular language.

- Recall the "convenient" properties of GNFA:
- start state goes to every other state, but has no incoming states
- every state goes to the unique accepting state, which is different from the starting state. The accepting state does not have any outgoing arrows.
- all other states have one transition to all other states, including themselves.

$\Rightarrow$ When removing X, we must only consider situations like this:

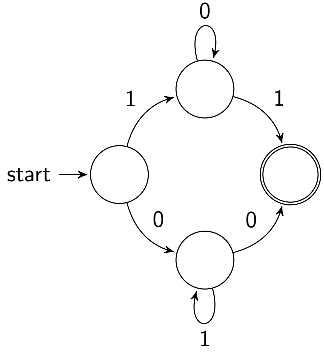


Proposition

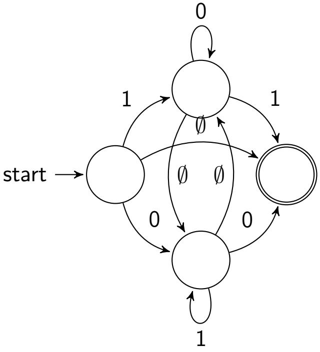
Every regular language can be described using a RE.

Example:

DFA:



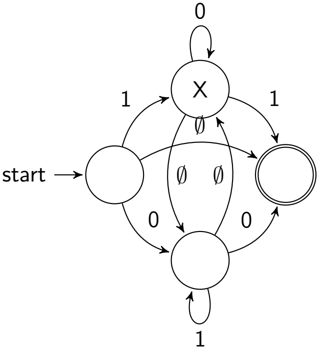
GNFA:

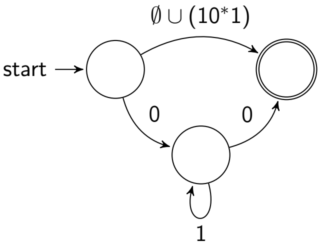


Proposition

Every regular language can be described using a RE.

Example:
Remove state X:



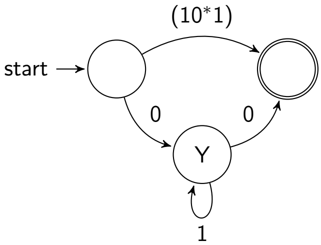


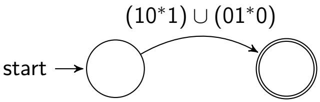
Regular Expressions and Automata

Proposition

Every regular language can be described using a RE.

Example:
Remove state Y:





Summary

So RE = GNFA = DFA = NFA = Regular languages...

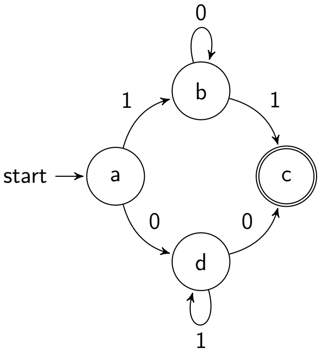
But when is a language *irregular*? How can we check? What tools have we seen?

$\Rightarrow$ Pumping Lemma!

Pumping Lemma

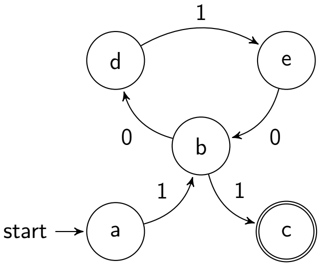
- DFAs only have *finite* memory, aka states.
- Pumping lemma gives a *pumping length*: if a string is longer than the pumping length, it can be *pumped*, i.e., there is a substring that can be repeated arbitrarily often such that the string remains in the language
- If a DFA as p states, and a string has length $\geq p$, then the accepting path in the DFA must visit at least $p + 1$ states. In other words, at least one state appears twice. $\Rightarrow$ loop!
- This loop can be repeated while staying in the language.

Pumping Lemma - Example



- Language $(10^*1) \cup (01^*0)$
- DFA has 4 states
- consider string 10001, length 5
- $\Rightarrow$ path must contain a loop (in this case, at node b)

Pumping Lemma - Example



- Language $1(010)^*1$
- DFA has 5 states
- consider string 10101, length 5
- $\Rightarrow$ path must contain a loop (in this case, at nodes b,d,e)
- $\Rightarrow$ 10100101 is also a word!

Pumping Lemma

Lemma (Pumping Lemma)

If A is a regular language, then there is a number p, called the pumping length, where if s is a word in A of length $\geq p$ then s can be divided into three parts, $s = xyz$, such that

1. $xy^iz \in A$ *for every* $i \geq 0$,
2. $|y| > 0$,
3. $|xy| \leq p$.

Pumping Lemma

- very useful for determining if a language is irregular
- $\rightarrow$ find a string with length $\geq$ p such that the pumping lemma does not hold
- *not* very useful for proving a language is regular
- $\rightarrow$ not an if and only if statement!

Pumping Lemma - Applied

Lemma (Pumping Lemma)

If A is a regular language, then there is a number p, called the pumping length, where if s is a word in A of length $\geq p$ then s can be divided into three parts, $s = xyz$, such that

1. $xy^iz \in A$ *for every* $i \geq 0$,
2. $|y| > 0$,
3. $|xy| \leq p$.

- Let $A = \{0^n1^n \mid n \geq 0\}$.
- Is A regular?
- If it is, then the pumping lemma gives us a pumping length p.
- Let $s = 0^p1^p$.

Pumping Lemma - Applied

Lemma (Pumping Lemma)

If A is a regular language, then there is a number p, called the pumping length, where if s is a word in A of length $\geq p$ then s can be divided into three parts, $s = xyz$, such that

1. $xy^iz \in A$ *for every* $i \geq 0$,
2. $|y| > 0$,
3. $|xy| \leq p$.

- Let $A = \{0^n1^n \mid n \geq 0\}$.
- Let $s = 0^p1^p$.
- Condition 3 tells us that y consists of only 0s.
- $\Rightarrow$ then xy^iz for $i \geq 2$ has more 0s than 1s. Contradiction! $\Rightarrow$ A is irregular.

Pumping Lemma - Applied

- Even if a language is irregular, it might contain strings for which the pumping lemma is true!
- We have to be careful!

Pumping Lemma - Applied

Lemma (Pumping Lemma)

1. $xy^iz \in A$ *for every* $i \geq 0$,
2. $|y| > 0$,
3. $|xy| \leq p$.

- Let $B = \{\omega \,|\, \omega$ contains an equal number of 0s and 1s$\}$.
- Let $s = (01)^p$.
- $x = \varepsilon, y = 01, z = (01)^{p-1}$
- all conditions are met!

Pumping Lemma - Applied

Lemma (Pumping Lemma)

1. $xy^iz \in A$ *for every* $i \geq 0$,
2. $|y| > 0$,
3. $|xy| \leq p$.

- Let $B = \{\omega \,|\, \omega$ contains an equal number of 0s and 1s$\}$.
- Let $s = 0^p1^p$.
- $x = \varepsilon, y = 0^p1^p, z = \varepsilon$
- looks like it can be pumped, but are all conditions met?
- condition 3 $\Rightarrow$ y must contain only 0s, so it cannot be pumped $\Rightarrow$ B irregular!

Pumping Lemma - Applied

- $A = \{0^n 1^n \mid n \geq 0\}$.
- $B = \{\omega \mid \omega \text{ contains an equal number of 0s and 1s}\}$
- Another way of showing B is irregular is to reduce it to the irregularity of A:
- regular languages are closed under intersection
- and $A = B \cap 0^* 1^*$
- if B is regular and since $0^* 1^*$ is regular, then A must be as well, contradiction!
- another way of saying this is: if a language contains an irregular language, it must be irregular as well!

Summary

- regular expressions are shorthand notations for languages
- RE = GNFA = DFA = NFA, i.e., regular expressions are shorthand for *regular* languages
- proof involved transforming a DFA to a GNFA then reducing the number of states to 2 while accepting the same language
- $\rightarrow$ the regular expressions describe the paths in the DFA
- every regular language has a pumping length
- useful for determining if a language is *irregular*